



Biotic Prediction

Building the Computational Technology Infrastructure
for Public Health and Environmental Forecasting

Configuration Management Plan

BP-CMP-1.0

Task Agreement: GSFC-CT-1

April 8, 2002

Contents

1 Overview	2
1.1 Introduction	2
1.2 Referenced Documents	2
1.3 Document Overview	2
2 Organization and Resources	3
2.1 Configuration Management System	3
2.2 Personnel, Roles and Responsibilities	3
3 Procedures, Tools and Records	4
3.1 Configuration Identification	4
3.2 Configuration Control	4
3.2.1 CVS Tags	4
3.3 Requirements Change Control	5
3.4 Configuration Status Accounting	5
3.5 Audits and Reviews	5
3.6 Storage and Handling	5
3.7 Delivery	6
A Versions	7
B Glossary	8

List of Tables

1 Referenced Documents	2
----------------------------------	---

List of Figures

1 Configuration Change Request Process	5
--------------------------------------------------	---

1 Overview

1.1 Introduction

This project will develop the high-performance, computational technology infrastructure needed to analyze the past, present, and future geospatial distributions of living components of Earth environments. This involves moving a suite of key predictive, geostatistical biological models into a scalable, cost-effective cluster computing framework; collecting and integrating diverse Earth observational datasets for input into these models; and deploying this functionality as a Web-based service. The resulting infrastructure will be used in the ecological analysis and prediction of exotic species invasions. This new capability will be deployed at the USGS Midcontinent Ecological Science Center and extended to other scientific communities through the USGS National Biological Information Infrastructure program.

1.2 Referenced Documents

Table 1. Referenced Documents

Document Title	Version	Date
Software Engineering / Development Plan	1.0	2002-04-08
Configuration Management Plan	1.0	2002-04-08

1.3 Document Overview

This document, the *Configuration Management Plan*, describes our plan for tracking the configuration of elements of the developed system and managing change throughout the lifecycle of the project.

Section 2 describes the Configuration Management organization within the project and the roles and responsibilities the personnel have with regard to Configuration Management.

Section 3 describes the procedures, tools, and records that will be used for Configuration Management.

Appendix A provides some guidelines that will govern the assignment of release versions to software configuration items, and appendix B has a glossary of terms and acronyms used in this document.

2 Organization and Resources

2.1 Configuration Management System

CVS, the *Concurrent Versions System*,¹ is a tool that will be used to store all versions of the software and to track changes and baselines for the project.

2.2 Personnel, Roles and Responsibilities

Project Manager (PM) The Project Manager has ultimate responsibility for all activities and will be informed and involved about major CM releases and decisions.

Configuration Manager (CM) The Configuration Manager will be responsible for overall management of the configuration management repository and identification of configuration items (CIs) to be managed.

Lead Developer Each CI will be assigned a Lead Developer who has primary responsibility for the management and direction of change for that CI.

Developer Each CI will be assigned to a Unix group and each developer who will be making changes to the CI will be a member of that Unix group. Each of the developers can check out, make changes, and commit changes to the CI as needed.

CCB A Configuration Control Board (CCB) comprised of the Project Manager, the Configuration Manager, and the Lead Developer for the CI under consideration will make decisions regarding major changes as needed.

Any changes to the requirements must be approved by the CCB.

¹<http://www.cvshome.org>

3 Procedures, Tools and Records

3.1 Configuration Identification

Configuration Management will extend to documents as well as project source code. Each document will be assigned a unique identifier and will be configured separately.

Project source code may be broken into separately configured items as appropriate. For example, routines that are useful apart from the main body of the code should be split into their own separately configured library. This separation could make the work more useful to the community.

Within the project configuration management repository (“CVSROOT”), each configuration item (CI) will be individually checked into CVS.

Each CI (document, source code, or other) will be assigned a short mnemonic identifier (CI name) that will be used by the CVS system to refer to that particular CI.

A list of all CIs, a short summary/description of the CI, and a “Lead Developer” for the CI will be maintained by the Configuration Manager (CM) and will be accessible on the project web page.

3.2 Configuration Control

Once it is time to place items under configuration control, the CM will assign the CI name and the item will be checked into the project CVS repository with that name.

During initial check in, the CI will be assigned a tag of “START” which can always be used to retrieve the initial version of the item.

Each CI will be assigned a Unix group and all developers who are working on that CI will be a member of that Unix group. The developers can check out the CI, make changes, and commit them back to the CVS repository. The developer will attempt to accurately describe the changes in the CVS log entered with each commit.

Developers are encouraged to unit test the work prior to commit and only commit the changes when the software as a whole is in a working state. When making a significant number of changes that require commits prior to the system working, the use of “branches” can help track the changes without impacting the work of other developers of the same CI.

Periodically, after major revisions, new features, etc., the item will undergo more significant CI level testing and will receive a new baseline CVS tag with a new version for the CI. A ChangeLog will list the version baselines of the CI by CVS tag and briefly summarize the changes to the CI represented by that baseline.

The Lead Developer for the CI will be responsible for updating the ChangeLog, assuring that adequate CI level testing has occurred, assigning the new version number (see appendix A), and creating the CVS tag for the new baseline.

3.2.1 CVS Tags

The following CVS tags have special meanings:

START This always represents the initial checked in version of the CI.

REL.xxx A released version of the code. See appendix A for guidance on version numbers and their representation as CVS tags.

3.3 Requirements Change Control

Any change that affects the baselined requirements must be submitted to the CCB as a Configuration Change Request (CCR). Figure 1 depicts the typical consideration of a change request. The CCB will analyze and consider each new change request. If the request is rejected, it may be reworked and resubmitted. If it is accepted, the requirements will be rebaselined and everyone potentially affected by the change will be informed.

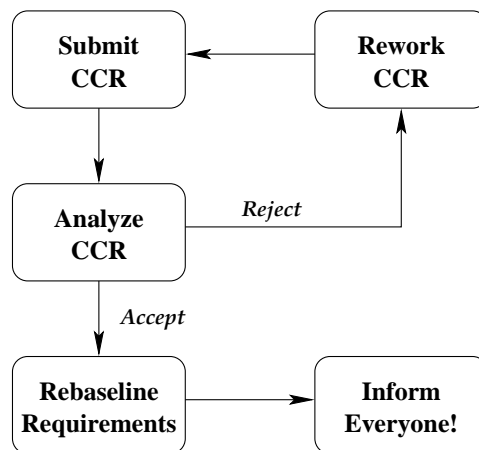


Figure 1. Configuration Change Request Process

A change in requirements will trigger updates to the design (and design documentation) and implementation as appropriate.

3.4 Configuration Status Accounting

Each time a major release occurs, the CM or Lead Developer will update the web accessible CI list with the version of the release. That page will always report the latest version of each CI.

3.5 Audits and Reviews

The CM will monitor the configuration of CIs and work with the Lead Developers to ensure that our configuration management procedures are being used appropriately.

Prior to major releases of the software, the Lead Developer will perform an audit of the configuration by establishing a “clean” area and checking out all the software from the CVS system, performing a complete build of the software from scratch, and re-running the test suite.

3.6 Storage and Handling

The CVS repository will be backed up nightly along with the rest of the system. Local access to the CVS repository will be governed by standard Unix permissions. Remote access to the CVS repository can take two forms:

- Read/write access by our developers will use the external SSH CVS method. This is subject to all the same login security and encryption that SSH itself uses.

- Read/only access by the public may be granted as an alternative to downloading release versions of the software for project-external users or developers.

Read/only access to the CVS repository through a web interface may also be established to aid in software development and maintenance.

3.7 Delivery

Periodically (see *Software Engineering / Development Plan* (SEP) for schedule details), the software will have a formal submission. The Lead Developer will ensure that the software to be submitted meets the criteria established for the delivery, perform the validation test suite as specified, and give the software a release tag.

The software, as tagged for the release, will also be packaged and placed on the public web site, clearly identified by version.

A Versions

Versions shall be represented as a list of integers. No words or letters should be used in version designations. These will typically be depicted by separating the integers with periods ('.'). The use of periods to separate the series of integers does not imply that they are fractions. The integers to the right of periods are integers just like the first integer.

The rank of the integer in the version represents a level of its contribution to the version as a whole. If the first integer of one version is higher than the first integer of a second version, then the whole version is higher. If the first integers are equal, then the second integers are likewise compared. For example: 1.0 is higher than 0.5 as expected. Unlike floating point numbers however, version 0.10 is higher than version 0.9. Version 0.2.10 is higher than 0.1.9 since the second integer is higher.

The establishment and management of versions is largely up to the developers, within the following constraints:

1. Later versions of the software will be represented by higher version numbers.
2. Because CVS reserves the use of periods in tags, our convention will be to replace them with underscores: i.e. to tag for version 0.5.2, use 0_5_2.
3. Certain threshold versions have been assigned special meaning and will not be exceeded until the software meets the criteria defined for those versions.

The following guidelines are recommended for consistency (as long as they do not violate the constraints above):

1. Increment the least significant integer of the version for minor patches that fix bugs. That is, if there is a minor problem with version 0.6, the version that patches that bug should be 0.6.1. If another minor problem is fixed, go to 0.6.2 and so on. The tenth such patch might be 0.6.10. For a minor fix to the patch for version 0.6.10, you might even go to version 0.6.10.1.
2. Increment more significant integers for major changes such as fundamental science algorithm changes or major software reorganizations (i.e., from 0.6.10 go to version 0.7.).

B Glossary

BP Biotic Prediction project

CCB Configuration Control Board

CCR Configuration Change Request

CI Configuration Item

CM Configuration Manager or Configuration Management

CMP Configuration Management Plan

CVS Concurrent Versions System

SSH Secure Shell